This listing of claims will replace all prior versions, and listings of claims in the application:

Listing of the Claims:

---

1. (Currently Amended) A method for assuring compatibility between a formal specification, a front-end debugger program, and a back-end debugger program which interfaces with a debuggee system, the method comprising:

inputting a formal specification into a code generator;

parsing the formal specification **utilizing the code generator**;

generating a front-end debugger program portion from the formal specification **based on the parsing of the formal specification**;

generating a back-end debugger program portion from the formal specification **based on the parsing of the formal specification**; and

wherein the front-end debugger program **portion** and back-end debugger program **portion** are compatible with each other **and comply with the formal specification**.

2. (Original) The method of Claim 1, wherein the front-end debugger program runs on a first virtual machine.

3. (Original) The method of Claim 2, wherein the front-end debugger program portion comprises Java programming language code.

4. (Previously Amended) The method of Claim 2, wherein the back-end debugger program directly controls and communicates with a second virtual machine.

5. (Original) The method of Claim 4, wherein the back-end debugger program portion comprises C language code.

6. (Original) The method of Claim 1, wherein the formal specification is a Java Debug Wire Protocol specification.

7. (Original) The method of Claim 6, further comprising generating HTML code that contains a human-readable description of the protocol specification.

8. (Previously Amended) The method of Claim 1 further comprising enabling a communication protocol between the front-end debugger program and the back-end

debugger program, wherein the communication protocol is defined by the formal specification.

9. (Original) The method of Claim 1, wherein the formal specification is written in a declarative language.

10. (Original) The method of Claim 8, wherein the communication protocol is a Java Debug Wire Protocol.

11. (Original) The method of Claim 8, further comprising generating HTML code from the formal specification that contains a human-readable description of the communication protocol defined by the formal specification.

12. (Previously Amended) A method for automatically generating front-end debugger interface code and back-end debugger agent interface code that are both compatible with a communication protocol, the method comprising:

creating a formal specification file that contains a description of a communication protocol between a front-end debugger code and a back-end debugger agent code;

inputting the formal specification file into a code generator;

utilizing the code generator to parse the formal specification;

generating the front-end debugger interface code from the formal specification;

generating the back-end debugger agent interface code from the formal specification; and

wherein the front-end debugger interface code and the back-end debugger agent interface code are compatible with each other.

13. (Original) The method of Claim 12, wherein the front-end debugger interface code comprises Java code, the back-end debugger agent interface code comprises C code, and the formal specification comprises a specification language.

14. (Original) The method of Claim 13, wherein the communication protocol is a Java Debug Wire Protocol.

15. (Currently Amended) A computer readable medium including computer program code for automatically generating front-end debugger interface code and back-end debugger interface code that are both compatible with a communication protocol, the computer readable medium comprising:

computer program code for inputting a formal protocol specification into a code generator;

computer program code for utilizing the code generator to parse the formal protocol specification;

computer code for generating front-end debugger interface computer code from the formal specification **based on the parsing of the formal specification**;

computer code for generating back-end debugger interface computer code from the formal specification **based on the parsing of the formal specification**; and

wherein the front-end debugger interface computer code and back-end debugger interface computer code are compatible with each other **and the formal specification**.

16. (Original) The medium of claim 15, further comprising computer code for generating HTML code containing a human-readable description of the communication protocol.

17. (Original) The medium of Claim 16, wherein the communication protocol is a Java Debug Wire Protocol.

18. (Currently Amended) A computer system for automatically generating front-end debugger interface code and back-end debugger interface code that are both compatible with a communication protocol, the computer system comprising:

a processor; and

a computer program operating on the processor that reads in a formal communication protocol specification, parses the specification, and generates front-end debugger interface code and back-end debugger interface code **based on the parsing of the specification**, such that the front-end debugger interface code and the back-end debugger interface code are fully compliant with the specification and compatible with each other.

19. (Previously Added) A method for implementing a Java Debug Wire Protocol (JDWP) for communication between a Java debugger application and a Virtual Machine,

the method comprising:

inputting a formal specification written in JDWP specification language into a code generator;

parsing the formal specification **using the code generator**;

generating a Java front-end debugger program portion from the formal specification after the parsing of the formal specification, the Java front-end debugger program running on a first virtual Machine;

generating a back-end Java debugger program portion from the formal specification, the back-end Java debugger program portion capable of controlling and communicating with a second Virtual machine, and

wherein the Java front-end debugger program and Java back-end debugger program are compatible with each other **and complying with the specification**.

20. (Currently Added) A method of providing a debugging environment in a computing environment that includes first and second virtual machines, the method comprising:

inputting a formal specification written in a debugging specification language into a program code generator, the formal specification defining a high level debugging communication protocol for communication between the first and second virtual machines;

parsing the formal specification using the program code generator;

automatically generating a front-end debugger program portion from the formal specification based on the parsing of the formal specification, the front-end debugger program running on a first virtual machine, the front-end debugger program portion corresponding to a platform independent programming language which provides a high level debugging interface which can be accessed by a debugger application operating on a first virtual machine;

automatically generating a back-end debugger program code portion from the formal specification based on the parsing of the formal specification, the back-end debugger program code portion implementing a virtual machine debugging interface which provides the capability to control and communicate with a second Virtual machine, the back-end debugger program code portion corresponding to a platform-specific programming language; and

wherein the front-end debugger program portion and the back-end debugger program code that are generated from the formal specification are compatible with each other and comply with the formal specification, thereby implementing the high-level communication protocol between the first and second virtual machines.

21. (Currently Added) A method as recited in claim 20, wherein the method further comprises:

providing a front-end processing module and a back-end processing module respectively for the front-end debugger program portion and the back-end debugger program code portion; and

wherein the front-end processing module and the back-end processing module can be utilized to implement a transport mechanism for communication between the front-end debugger program portion and the back-end debugger code portion.

22. (Currently Added) A method as recited in claim 21, wherein the transport mechanism can be a socket, or a serial line, or socket, or a shared memory implementation.

23. (Currently Added) A method as recited in claim 20,

wherein the formal specification is written in a purely declarative language and,

wherein the formal specification defines the format of information and requests between the front-end debugger program portion and the back-end debugger program code portion, and

wherein the formal specification does not define the transport mechanism which is implemented for communication between the front-end debugger program and the back-end debugger code portion.

24. (Currently Added) A method as recited in claim 20, wherein the front-end processing module operates to send events that are generated in the second virtual machine to the front-end debugger program portion via the back-end debugger program code portion.

25. (Currently Added) A method as recited in claim 24, wherein the front-end processing module performs one or more of the following operations:

read and parse events from the back-end debugger code portion;

convert the events from a first format into a second format which is compatible with the front end debugger code portion; and

queue the events.

26. (Currently Added) A method as recited in claim 24, wherein the front-end processing module further performs operations related to requests made through the front-end debugger program by the debugger application program.

27. (Currently Added) A method as recited in claim 24, wherein the front-end processing module further performs one or more of the following operations:

write formatted requests

send the formatted requests to the back-end debugger code portion;

associate at least one reply with the formatted requests;

read and parse the at least one reply;

deliver the at least one reply to an appropriate requester.

28. (Currently Added) A method as recited in claim 27, wherein the back-end processing module performs operations related to event processing and request processing.

29. (Currently Added) A method as recited in claim 28, wherein the event processing operations performed by the back-end processing module includes sending an event which was generated through the virtual machine debugging interface to the front-end debugging portion.

30. (Currently Added) A method as recited in claim 29, wherein the request processing operations performed by the back-end processing module include one or more of the following operations:

reading and parsing formatted requests from the front-end debugger program portion;

forwarding the requests to the back-end debugger program code portion;

sending the reply to the requests to the front-end debugger program portion.

31. (Currently Added) A method as recited in claim 20, wherein the back-end processing module performs operations related to event processing and request processing.

32. (Currently Added) A method as recited in claim 20, wherein the front-end debugger program portion includes a class which is used by the front-end debugger program portion to send and receive information over the debugging communication protocol.